Module 5: Effect of Agile on Engineering & Test Staff

TLO: Given a DoD program involved in software development using Agile IPM methodologies, the student will understand differences in engineering and test roles to enable program success.

ELOs

1. Identify how program technical requirements are managed in Agile contracting settings
2. Identify how program baselines are managed in an Agile environment
3. Recognize key factors for developmental testing success in an Agile environment
4. Identify different ways that System Engineering Technical processes interact with Agile Software approaches

Assessment

LP - In Agile settings, low level requirements add who and why to the typical what of requirements, often using a format called a “story” (ELO 1)

LP - Requirements in an Agile contracting setting need to be prioritized in terms of the rank order of their value to the end user and other stakeholders (ELO 1)

MT - A requirements baseline that is at too low a level of abstraction is unproductive for an Agile contract setting (ELO 1)

MT - A capability-based Work Breakdown Structure makes developing and refining requirements in an Agile setting easier than using the more hardware-focused, but typical, component-based WBS (ELO 1)

MT - automated testing is a necessity, not an option, for any decent-sized Agile program (ELO 3)

Module Story

What story do we want to tell as motivation and to support terminal learning objective

Module Contents

1 - Systems Engineering – Requirements (ELO 1~~, 2~~)

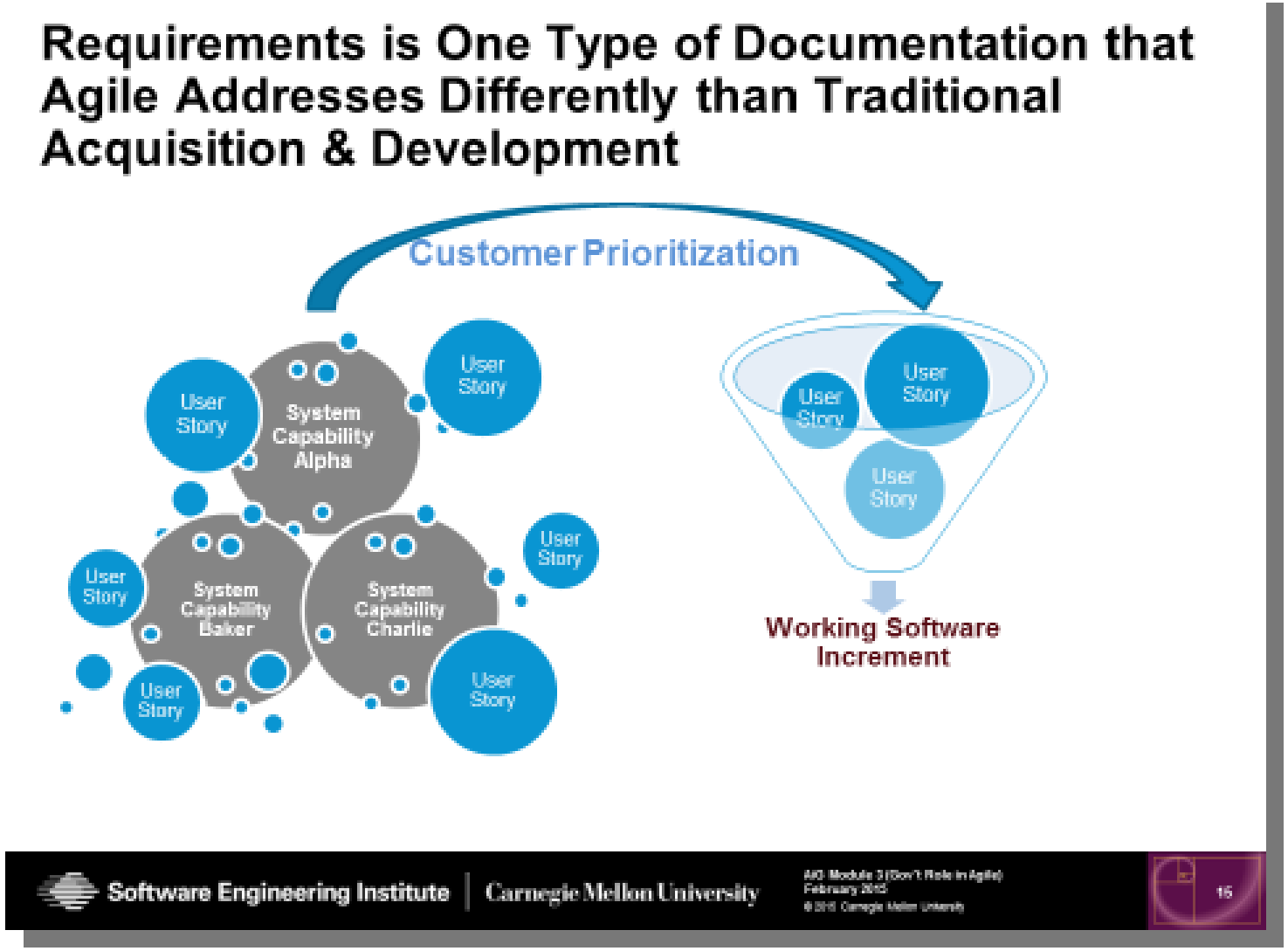
2 - Systems Engineering – Technical Processes (ELO 2)

3 - Integration and Testing (ELO 3)

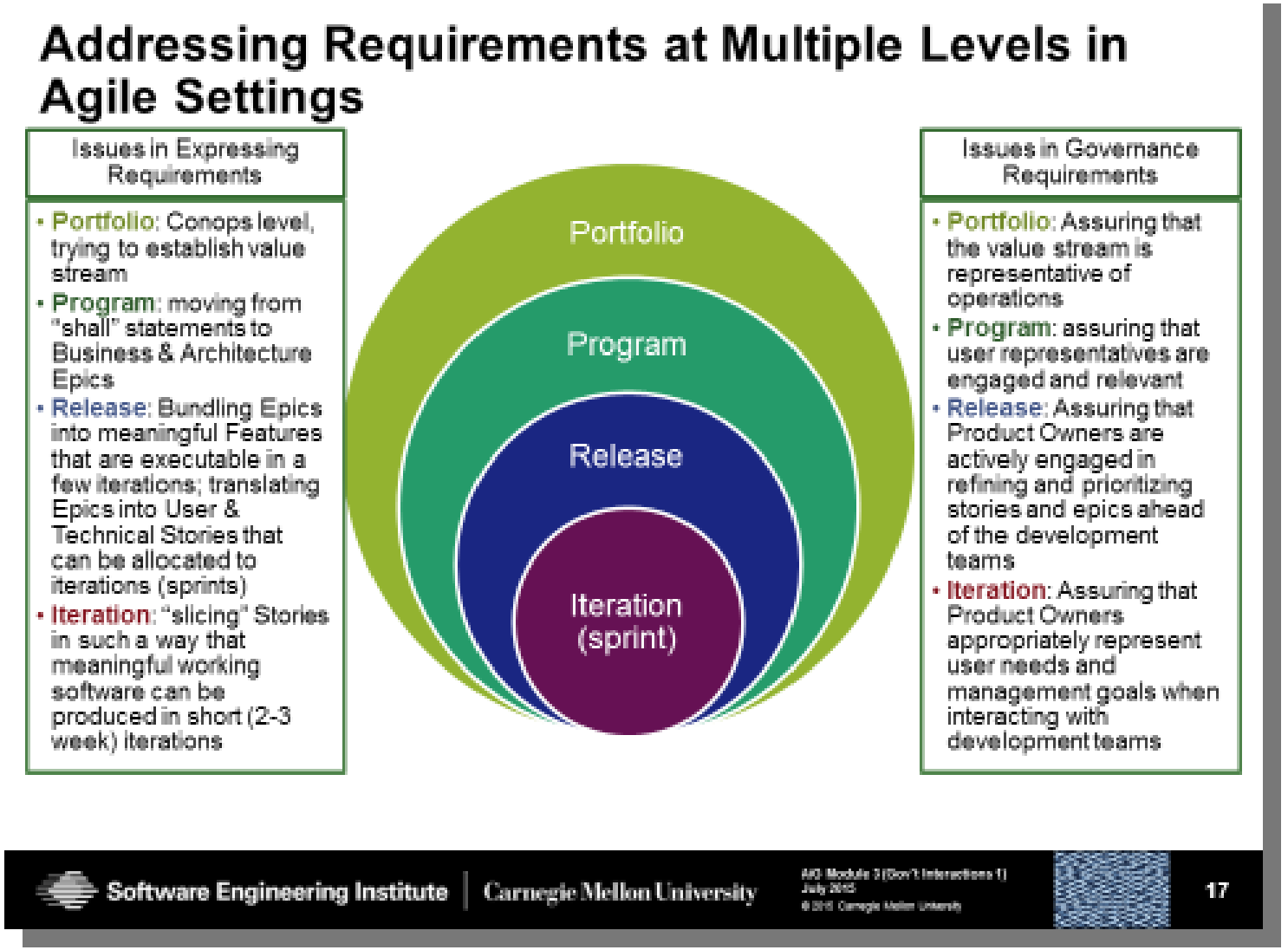
Subtopic 1: Systems Engineering - Requirements

- Requirements and Configuration Management
  - Flow of expressing requirements (somewhat of a review from Module 4)
    - Using a user story to flesh out details of requirements
    - Horizontal and vertical traceability
  - Flow and configuration management of requirements, prioritization and releases
- Major Takeaways
  - A requirements baseline that is at too low a level of abstraction is unproductive for an Agile contract setting.
  - In Agile settings, low level requirements add who and why to the typical what of requirements, often using a format called a “story”
  - A capability-based Work Breakdown Structure makes developing and refining requirements in an Agile setting easier than using the more hardware-focused, but typical, component-based WBS
  - Requirements in an Agile contracting setting need to be prioritized in terms of the rank order of their value to the end use and other stakeholders.

Suggested Content



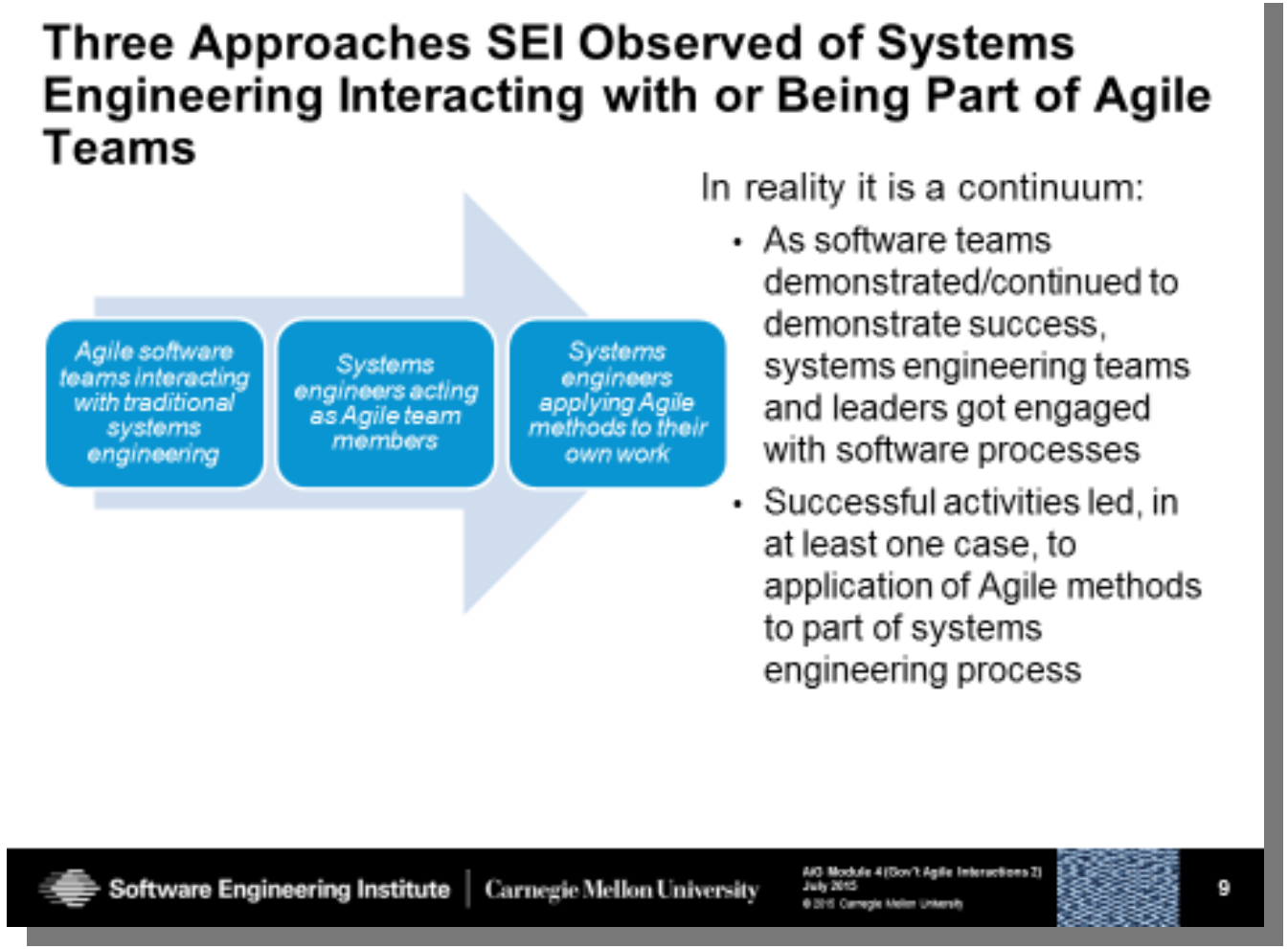
Suggested Content


Addressing Requirements at Multiple Levels in Agile Settings
Issues in Expressing Requirements
• Portfolio: Conops level, trying to establish value stream
• Program: moving from "shall" statements to Business & Architecture Epics
• Release: Bundling Epics into meaningful Features that are executable in a few iterations; translating Epics into User & Technical Stories that can be allocated to iterations (sprints)
• Iteration: "slicing" Stories in such a way that meaningful working software can be produced in short (2-3 week) iterations
Portfolio
Program
Release
Iteration (sprint)
Issues in Governance Requirements
• Portfolio: Assuring that the value stream is representative of operations
• Program: assuring that user representatives are engaged and relevant
• Release: Assuring that Product Owners are actively engaged in refining and prioritizing stories and epics ahead of the development teams
• Iteration: Assuring that Product Owners appropriately represent user needs and management goals when interacting with development teams
Software Engineering Institute | Carnegie Mellon University
17

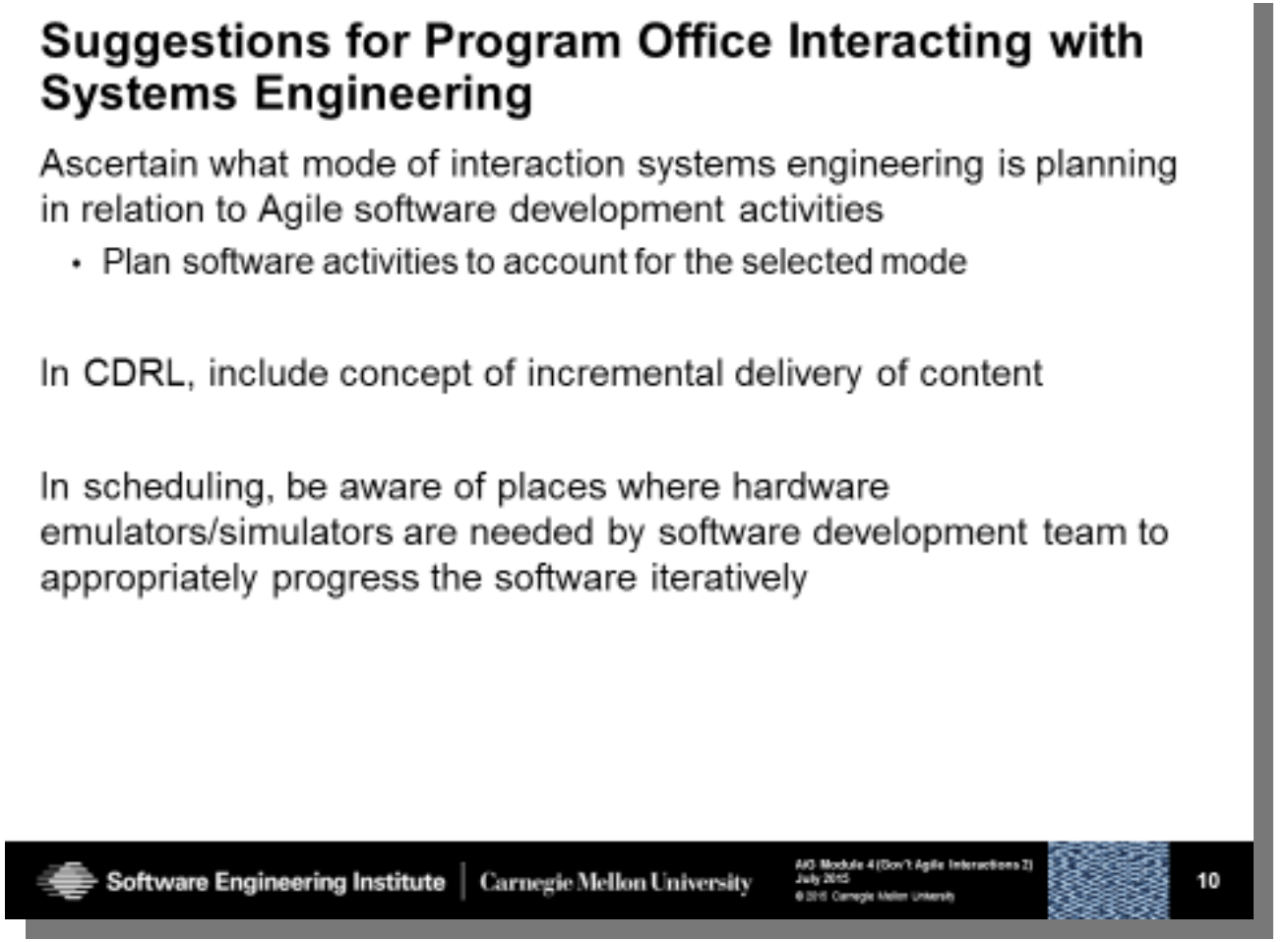
Subtopic 2: Systems Engineering – Technical Processes

- Approaches to managing interaction with Agile Software teams
  - Systems Engineers Acting as Agile Product Owner
  - Systems Engineers Acting as Agile Systems Architect
  - Systems Engineers Applying Agile Methods to Their Own Work
- Program baselines in an Agile setting (needs major work)
  - Technical reviews that establish and evolve program baselines
  - Functional, Allocated, Product
- Resource: Lean Engineering reference

Suggested Content


Three Approaches SEI Observed of Systems Engineering Interacting with or Being Part of Agile Teams
Agile software teams interacting with traditional systems engineering
Systems engineers acting as Agile team members
Systems engineers applying Agile methods to their own work
In reality it is a continuum:
As software teams demonstrated/continued to demonstrate success, systems engineering teams and leaders got engaged with software processes
Successful activities led, in at least one case, to application of Agile methods to part of systems engineering process
Software Engineering Institute | Carnegie Mellon University
9

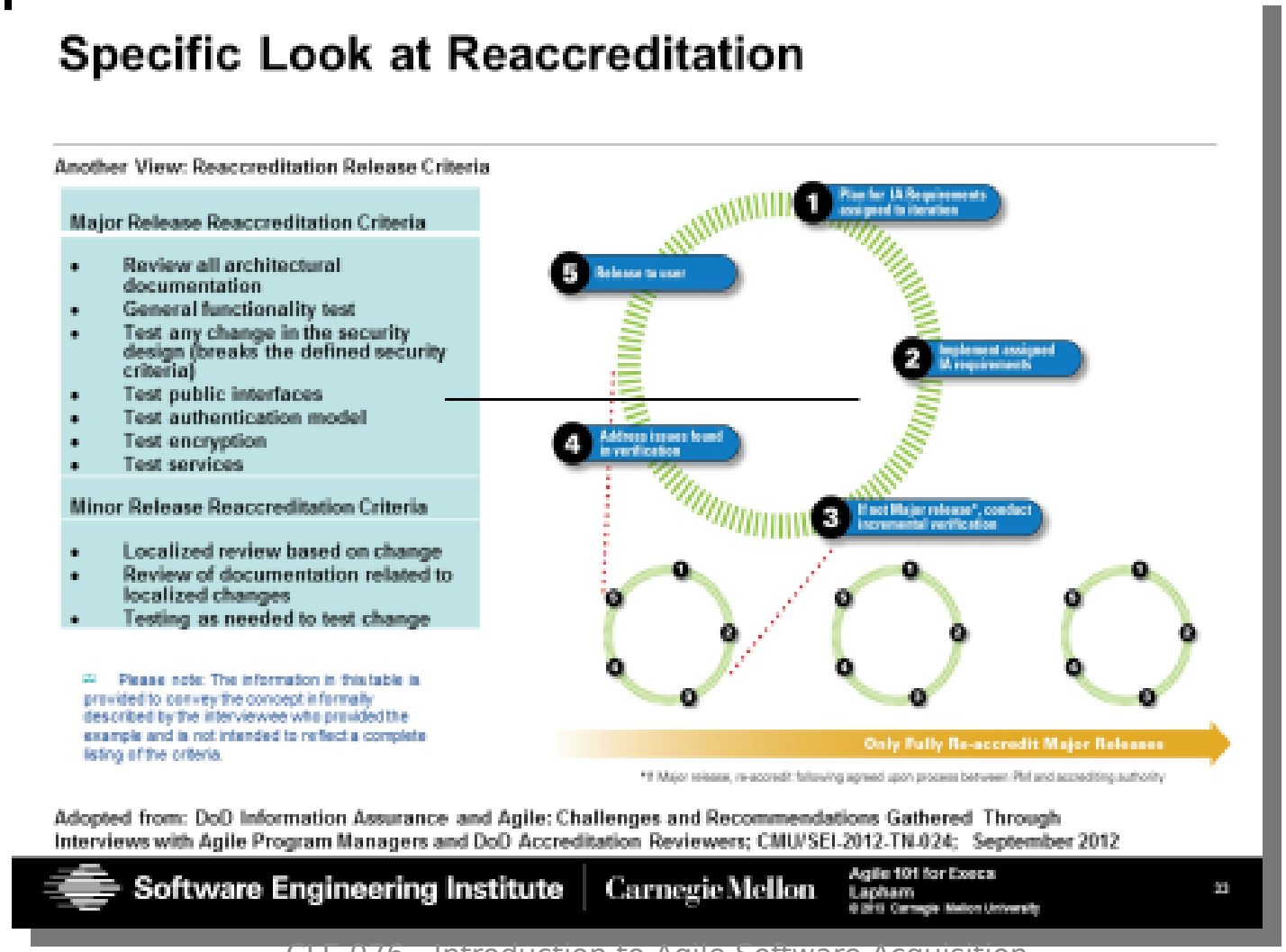
Suggested Content


Suggestions for Program Office Interacting with Systems Engineering
Ascertain what mode of interaction systems engineering is planning in relation to Agile software development activities
• Plan software activities to account for the selected mode
In CDRL, include concept of incremental delivery of content
In scheduling, be aware of places where hardware emulators/simulators are needed by software development team to appropriately progress the software iteratively
Software Engineering Institute | Carnegie Mellon University
10

Subtopic 3: Integration and Testing

- Use of supplemental test strategy to compliment the high level program TEMP
  - Deal with Measures of Effectiveness for traceability
  - Integration approaches for DT/OT activities
- Developmental testing and evaluation in an iterative approach
  - Cybersecurity staff involvement
  - Ensuring integrity of the definition of done – including cyber
  - Leverage multiple sources of evidence (unit testing, demos, traditional system testing)
- Automated testing and automation support
- Major Takeaways
  - automated testing is a necessity, not an option, for any decent-sized Agile program

Suggested Content—Need for Reaccreditation


Specific Look at Reaccreditation
Another View: Reaccreditation Release Criteria
Major Release Reaccreditation Criteria
Review all architectural documentation
General functionality test
Test any change in the security design (breaks the defined security criteria)
Test public interfaces
Test authentication model
Test encryption
Test services
Minor Release Reaccreditation Criteria
Localized review based on change
Review of documentation related to localized changes
Testing as needed to test change
Please note: The information in this table is provided to convey the concept informally described by the interviewee who provided the example and is not intended to reflect a complete listing of the criteria.
1
2
3
4
5
Only Fully Re-accredit Major Releases
Adopted from: DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers; CMU/SEI-2012-TN-024; September 2012
Software Engineering Institute
Carnegie Mellon
Agile 101 for Execs
Lapham

Suggested Content—Cybersecurity recommendations

Certification and Accreditation Recommendations
Define criteria for reaccreditation early in the project.
Leverage long accreditation approval wait time with frequent community previews.
Ensure requirements are prioritized according to value and risk.
Make sure Agile project teams understand the intent behind security requirements.
Ensure Agile development processes produce "just enough" security architecture.
Make sure there is at least one person with strong security analysis expertise on the team.
Foster Agile project team and accrediting authority collaboration.
Don't apply all the information assurance controls blindly.
Use COEs, SDKs and enterprise services to reduce accreditation time.
Apply a risk-based, incremental approach to security feature design and development.
Leverage architecture tactics such as layering and encapsulation to minimize impact of change.
Leverage unclassified environments for development and community previews.
Encourage the Agile teams to merge Agile and security best practices.
Software Engineering Institute
Carnegie Mellon
Agile 101 for Execs
Lapham
© 2013 Carnegie Mellon University
34